Triangle Test Protocol v1.0

Multi-Validator Consensus for AI Output Validation

A Framework for Ensuring AI Reliability Through Structured Validation

****Timothy I. Wheels****

Contruil LLC

December 2024

Executive Summary

The Triangle Test Protocol is a three-validator consensus system that validates AI outputs before delivery through four sequential validation gates. Unlike single-validator approaches, it separates factual validation from structural validation, creating redundancy and catching errors that any single validator might miss.

Key Innovation: Three distinct validators (Guardian, Perplexity, Claude Architecture) working in parallel and sequence, with different validation scopes—sovereignty, factual accuracy, and structural coherence.

The Problem: Why Single Validators Fail

Most AI validation systems use a single validator to check outputs. This creates blind spots:

- **Factual errors** slip through when structural checks pass
- **Structural problems** go unnoticed when facts are correct
- **Sovereignty violations** aren't caught by content-focused validators
- **No redundancy** means one validator's blind spot becomes a system failure

Example: An AI output might be factually accurate and well-structured, but violate user sovereignty by making decisions the user should make. A single validator focused on content quality won't catch this.

The Solution: Triangle Test Protocol

The Triangle Test uses **three validators** working across **four sequential gates**, each checking different aspects of the output:

1. **Guardian VLAN 99** — Sovereignty and policy enforcement
2. **Perplexity** — Factual accuracy and citation verification
3. **Claude Architecture** — Structural coherence and logical flow

Why "Triangle"?

The three validators form a triangle of validation—each checking different dimensions. Like a triangle needs three points to be stable, the protocol needs three validators to be reliable.

The Four-Gate Validation Framework

Gate 1: Meta-Analysis

Validator: Guardian VLAN 99

Purpose: Sovereignty preservation

Checks:

• Does the output preserve user sovereignty?

• Does it claim decision-making authority it shouldn't have?

• Are user boundaries respected?

Failure Action: **IMMEDIATE HALT**

Gate 1 failures are critical—if sovereignty is violated, the output must not be delivered.

Example Failure: Output says "I've decided to..." instead of "You might consider..."

Gate 2: Routing Verification

Validator: Guardian VLAN 99

Purpose: ACL compliance and routing accuracy

Checks:

• Was the query served by the correct VLAN?

• Were access control lists (ACLs) respected?

• Does the routing match the intended context?

Failure Action: ESCALATION (with option to re-route)

Example Failure: Technical query routed to narrative layer, or sensitive query routed to public context.

Gate 3: Payload Assessment

Validators: Perplexity (3a) + Claude Architecture (3b)

Purpose: Parallel validation of factual and structural quality

Gate 3a — Factual Accuracy (Perplexity):

• Are claims factually correct?

• Are citations valid and current?

• Can information be verified from sources?

Gate 3b — Structural Coherence (Claude Architecture):

• Is the narrative flow logical?

• Is technical depth appropriate?

• Are arguments well-structured?

Failure Action: ESCALATION (content can be fixed without full restart)

Key Innovation: Parallel execution means both validators run simultaneously, catching different types of errors without blocking each other.

Gate 4: Audit Trail Integrity

Validator: Guardian VLAN 99

Purpose: Evidence chain verification

Checks:

• Is the evidence chain complete and valid?

• Are all validation steps logged?

• Can the process be audited?

Failure Action: ESCALATION (audit requirements not met)

Special Feature: For RED threat posture, Gate 4 includes cryptographic signatures to ensure tamper-proof audit trails.

Threat Postures: Context-Aware Validation

The protocol adapts validation intensity based on threat posture:

GREEN (Normal)

• Standard validation sequence

• All gates must PASS

• No cryptographic signatures required

AMBER (Elevated)

• Enhanced scrutiny

• Missing validators trigger AWAITING_CLARIFICATION

• Requires operator review before delivery

RED (Critical)

• Maximum validation intensity

• Cryptographic signatures on Gate 4

• Mandatory operator approval

• Full audit trail with cryptographic proofs

Implementation: How It Works

Sequential Execution

Gates execute in order: 1 → 2 → 3 (parallel) → 4

Early Termination:

• Gate 1 failure → immediate halt (no further gates execute)

• Gate 2 failure → escalation, but Gate 3 may still run for context

• Gate 3 failure → escalation, Gate 4 still runs for audit trail

Parallel Validation (Gate 3)

Both Gate 3a and Gate 3b execute simultaneously:

Graceful Degradation

If a validator is unavailable:

• **Guardian missing** → Gate 1 fails (hard requirement)

• **Perplexity missing** → Gate 3a → AWAITING_CLARIFICATION

• **Claude missing** → Gate 3b → AWAITING_CLARIFICATION

In AMBER/RED postures, missing validators trigger escalation. In GREEN posture, the system can proceed with available validators.

Validation Results

Each gate returns one of three results:

PASS

■ Output meets validation criteria

Action: Continue to next gate

FAIL

■ Output fails validation criteria

Action: Escalate to operator (or halt if Gate 1)

AWAITING_CLARIFICATION

■■ Validator unavailable or needs more context

Action: Escalate or proceed based on threat posture

Real-World Example

Scenario: Technical Documentation Request

Input: "Explain how to implement OAuth2 authentication"

Gate 1 (Guardian): ■ PASS

• No sovereignty violations

• Output doesn't claim decision authority

Gate 2 (Guardian): ■ PASS

• Query correctly routed to technical/architect layer

• ACLs respected (user has access to technical content)

Gate 3a (Perplexity): ■ PASS

• OAuth2 facts are accurate

• OAuth2 flow correctly explained

• Citations to RFC 6749 are valid

Gate 3b (Claude Architecture): ■ PASS

• Logical flow: Introduction → Concepts → Implementation

• Technical depth appropriate

• Code examples well-structured

Gate 4 (Guardian): ■ PASS

• All validation steps logged

• Evidence chain complete

• Audit trail valid

Result: **DELIVER TO USER**

Output approved by all three validators across four gates.

Scenario: Failed Validation

Input: "What should I do about my medical condition?"

Gate 1 (Guardian): ■ PASS

• Sovereignty preserved (output suggests, doesn't decide)

Gate 2 (Guardian): ■ PASS

• Routed to appropriate layer

Gate 3a (Perplexity): ■ FAIL

• Output included medical claims not properly cited

• Factual accuracy concerns identified

Gate 3b (Claude Architecture): ■ PASS

• Structurally sound

Gate 4 (Guardian): ■ PASS

• Audit trail intact

Result: **ESCALATE TO OPERATOR**

Factual accuracy concerns require human review before delivery.

Key Design Principles

1. Separation of Concerns

Each validator checks what it's best at:

• Guardian → Policy and sovereignty

• Perplexity → Facts and citations

• Claude → Structure and logic

2. Defense in Depth

Multiple validators catch different types of errors:

• One validator's blind spot is another's strength

• Redundancy increases reliability

3. Fail-Safe Defaults

• Gate 1 failures halt immediately (safety first)

• Missing critical validators trigger escalation

• RED posture requires maximum validation

4. Evidence-Based Decisions

• Every validation step is logged

• Audit trails enable post-mortem analysis

• Cryptographic proofs for critical decisions

Benefits Over Single-Validator Systems

Aspect	Single Validator	Triangle Test
Error Detection	Limited to validator's scope	Cross-validator redundancy
Blind Spots	Entire system vulnerable	Multiple perspectives catch issues
Factual Accuracy	May miss citation errors	Perplexity specializes in verification
Structural Quality	May miss logical issues	Claude Architecture validates structure
Sovereignty	Often overlooked	Guardian explicitly checks

Audit Trail	Single point of failure	Distributed evidence chain

Implementation Considerations

Performance

• Parallel Gate 3 validation minimizes latency

• Early termination on Gate 1 failures saves resources

• Graceful degradation maintains service availability

Scalability

• Validators can be scaled independently

• Gate 3 parallel execution reduces bottlenecks

• Evidence logging can be async

Reliability

• Redundancy across validators

• Fail-safe defaults (halt on critical failures)

• Audit trails enable debugging and improvement

Use Cases

The Triangle Test Protocol is ideal for:

1. **AI Content Generation Systems**

• Ensure outputs meet quality and policy standards

• Catch factual errors before delivery

• Maintain user sovereignty

2. **Enterprise AI Applications**

• Compliance with internal policies
• Audit requirements
• Risk mitigation

3. **Consumer AI Products**
• Quality assurance
• Safety checks
• User trust through transparency

4. **High-Stakes AI Systems**
• Medical, legal, financial contexts
• Maximum validation for critical decisions
• Cryptographic audit trails

Next Steps: Implementing Triangle Test

Phase 1: Core Infrastructure

• Set up Guardian validator
• Integrate Perplexity API
• Integrate Claude API
• Build gate execution framework

Phase 2: Gate Implementation

• Implement Gate 1 (Meta-Analysis)
• Implement Gate 2 (Routing Verification)
• Implement Gate 3 (Payload Assessment)
• Implement Gate 4 (Audit Trail)

Phase 3: Integration

• Connect to your AI system
• Configure threat postures

• Set up escalation workflows
• Build operator review interface

Phase 4: Production

• Monitor validation results
• Tune validator thresholds
• Optimize performance
• Expand audit capabilities

Technical Specifications

Correlation IDs

Every validation run receives a unique correlation ID:

• Format: `triangle-YYYYMMDDTHHMMSSffffffZ`
• Links all gates together
• Enables full audit trail reconstruction

Timestamps

All validator approvals require ISO-8601 timestamps:

• UTC with milliseconds
• Format: `2024-12-31T12:34:56.789Z`
• Enables temporal analysis

Evidence Chain

Gate 4 validates the complete evidence chain:

• Links all gate results
• Includes correlation ID
• Cryptographic signatures for RED posture

Conclusion

The Triangle Test Protocol provides a robust framework for validating AI outputs through structured, multi-validator consensus. By separating concerns (sovereignty, facts, structure) and using parallel validation, it catches errors that single-validator systems miss.

Key Takeaways:

1. **Three validators** working together are more reliable than one
2. **Four sequential gates** ensure comprehensive validation
3. **Separation of concerns** means each validator checks what it's best at
4. **Fail-safe defaults** prioritize safety over speed
5. **Evidence-based** approach enables continuous improvement

About the Author

Timothy I. Wheels is a systems thinker and builder focused on AI reliability, systems architecture, and project execution. He translates chaos into frameworks, and frameworks into outcomes.

Learn more:

• Website: timothywheels.com

• LinkedIn: linkedin.com/in/timothywheelspro

• Company: contruil.com

Patent Status

The Triangle Test Protocol v1.0 is patent-pending (Q1 2026).

Status: FROZEN FOR LEGAL REVIEW (2024-12-23)

Entity: Contruil LLC

Version History

v1.0 (2024-12-23)

• Initial frozen specification

• Four-gate validation framework

• Three-validator consensus system

• Threat posture support

• Production implementation ready

Visual Elements (For PDF Design)

Placeholder 1: Triangle Test Flow Diagram

[INSERT: Flowchart showing: Input → Gate 1 → Gate 2 → Gate 3 (Parallel) → Gate 4 → Output/Escalation]

Placeholder 2: Three-Validator Triangle

[INSERT: Triangle diagram with Guardian, Perplexity, Claude at vertices, showing validation scope]

Placeholder 3: Gate Decision Tree

[INSERT: Decision tree showing PASS/FAIL/AWAITING paths for each gate]

Placeholder 4: Threat Posture Comparison

[INSERT: Table or visual showing GREEN/AMBER/RED differences]

End of Document